



# Schedulability Analysis for non Necessarily Harmonic Real-Time Systems with Precedence and Strict Periodicity Constraints using the Exact Number of Preemptions and no Idle Time

Patrick Meumeu Yomsi, Yves Sorel

## ► To cite this version:

Patrick Meumeu Yomsi, Yves Sorel. Schedulability Analysis for non Necessarily Harmonic Real-Time Systems with Precedence and Strict Periodicity Constraints using the Exact Number of Preemptions and no Idle Time. [Research Report] RR-6610, INRIA. 2008, pp.21. inria-00310248v2

**HAL Id: inria-00310248**

**<https://inria.hal.science/inria-00310248v2>**

Submitted on 11 Nov 2008

**HAL** is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE

***Schedulability Analysis for non Necessarily  
Harmonic Real-Time Systems with Precedence and  
Strict Periodicity Constraints using the Exact  
Number of Preemptions and no Idle Time***

Patrick Meumeu Yomsi — Yves Sorel

**N° 6610**

Août 2008

\_\_\_\_\_ Thème COM \_\_\_\_\_



# Schedulability Analysis for non Necessarily Harmonic Real-Time Systems with Precedence and Strict Periodicity Constraints using the Exact Number of Preemptions and no Idle Time

Patrick Meumeu Yomsi , Yves Sorel

Thème COM — Systèmes communicants  
Équipe-Projet AOSTE

Rapport de recherche n° 6610 — Août 2008 — 21 pages

**Abstract:** In this report we study hard real-time systems composed of dependent strictly periodic preemptive tasks in the monoprocessor case. Although preemptive scheduling algorithms are able to successfully schedule some systems that cannot be scheduled by any non preemptive scheduling algorithm, the cost of preemption may not be negligible. Therefore, its exact cost has to be explicitly considered in the schedulability conditions in order to avoid wasting resources and provide safety in terms of guaranteeing the right behavior of the system at run-time. Because we are interested in hard real-time systems with precedence and strict periodicity constraints where it is mandatory to satisfy these constraints, we have already shown in a previous work how to tackle this problem for systems composed of harmonic tasks. Two main contributions are presented in this report. First, we generalize our previous results to the case of systems with periods that are not necessarily harmonic. Second, we provide a necessary and sufficient schedulability condition which takes into account the exact number of preemptions for a system with such constraints when no idle time is allowed.

**Key-words:** schedulability analysis, scheduling algorithm, real-time systems, exact number of preemptions, exact cost of preemption, precedence constraint, strict periodicity constraint.

??

**Résumé :** Dans ce rapport nous étudions le problème d’ordonnancement dans les systèmes en temps réel durs composés de tâches dépendantes strictement périodiques dans le cas monoprocesseur. Bien que les algorithmes d’ordonnancement préemptifs soient en mesure d’ordonner certains systèmes ne pouvant être ordonnancés par n’importe quel algorithme d’ordonnancement non préemptif, le coût de la préemption peut ne pas être négligeable. Par conséquent, son coût exact doit être explicitement pris en compte dans les conditions d’ordonnancabilité afin d’éviter les gaspillages de ressources et de fournir la garantie du bon fonctionnement du système lors de son exécution. Nous nous intéressons, dans ce rapport, à des systèmes temps réel durs ayant des contraintes de précédence et de périodicité stricte, où il est obligatoire de satisfaire ces contraintes. Nous avons déjà montré dans un précédent travail comment aborder ce problème pour les systèmes composés de tâches harmoniques. Deux principales contributions sont présentées dans ce rapport. Premièrement, nous généralisons nos précédents résultats au cas des systèmes temps réel où les périodes des tâches ne sont pas nécessairement harmoniques. Deuxièmement, nous fournissons une condition d’ordonnancabilité nécessaire et suffisante qui prend en compte le nombre exact de préemptions pour un système avec de telles contraintes lorsque les temps creux ne sont pas autorisés.

**Mots-clés :** condition d’ordonnancabilité, algorithme d’ordonnancement, systèmes temps réel, coût exact de la préemption, nombre exact de préemptions, contrainte de précédence, contrainte de périodicité stricte.

# 1 Introduction

Scheduling theory as it applies to hard real-time environments with precedence and strict periodicity constraints — environments where the failure to satisfy any constraint may have disastrous consequences [1, 2, 3, 4] — seems currently to be enjoying a renaissance. The most widely studied problems concern domains such as automobiles, avionics, mobile robotics, telecommunications, etc, and concern periodic non preemptive tasks [5, 6, 7]. Although preemptive scheduling algorithms are able to successfully schedule some systems that cannot be scheduled by any non preemptive scheduling algorithm, the cost of preemption may not be negligible. Therefore, when preemption is allowed, its exact cost has to be explicitly considered in the schedulability conditions in order to avoid wasting resources and provide safety in terms of guaranteeing the right behavior of the system at run-time. In this report, we address the scheduling problem of hard real-time systems composed of **dependent, strictly periodic**, preemptive tasks in the monoprocessor case. The strictly periodic constraint implies that, for such a system, any task starts its execution at the beginning of its period whereas the dependence constraint implies that any task cannot start its execution before the end of another task preceding it. We assume here that no jitter is allowed at the beginning of each task. To clearly distinguish between the specification level and its associated model, we shall use the term *operation* rather than the commonly used “*task*” [8] which is too closely related to the implementation level.

For systems with the above-mentioned constraints, in [9] we proved that some of them can be eliminated because they are definitely not schedulable, then we solved the problem for systems with harmonic periods <sup>1</sup> in [10]. Here, we first generalize these results to the case of systems with periods that are not necessarily harmonic. Then, we provide a necessary and sufficient schedulability condition which takes into account the exact number of preemptions for a system with such constraints when no idle time is allowed. That means the processor always executes an operation if there is one to execute. Indeed, even though the cost  $\alpha$  of one preemption — the context switching time including the storage as well as the restoration of the context that the processor needs when a preemption occurs — is easy to know for a given processor, it remains a challenging problem to count the exact number of preemptions of each instance for a given operation [11, 12, 10]. As in [13], we consider only predictable processors without cache or complex internal architecture. We consider a set of  $n$  strictly periodic preemptive operations  $\tau_i$ ,  $1 \leq i \leq n$  with precedence constraints. Each operation  $\tau_i$  is an infinite sequence of instances <sup>2</sup>  $\tau_i^k$ ,  $k \in \mathbb{N}^+$ , and is characterized by a Worst Case Execution Time (WCET)  $C_i$ , not including any approximation of the cost of preemption, as is usually the case in the classical real-time scheduling theory [14, 15, 16, 17], and a period  $T_i$ . Regarding the constraints, we have the following information.

The **precedence** constraint is given by a partial order on the execution of the operations. An operation  $\tau_i$  preceding an operation  $\tau_j$  is denoted by  $\tau_i \prec \tau_j$  which means that  $s_i^k \leq s_j^k$ ,  $\forall k \geq 0$  thanks to the result given in [6]. In that paper it was proved that given two operations  $\tau_i = (C_i, T_i)$  and  $\tau_j = (C_j, T_j)$ :

$$\tau_i \prec \tau_j \implies T_i \leq T_j$$

<sup>1</sup> A sequence  $(a_i)_{1 \leq i \leq n}$  is harmonic if and only if there exists  $q_i \in \mathbb{N}$  such that  $a_{i+1} = q_i a_i$ . Notice that we may have  $q_{i+1} \neq q_i \quad \forall i \in \{1, \dots, n\}$ .

<sup>2</sup> Throughout the report all subscripts refer to operations whereas all superscripts refer to instances.

Since for two operations  $\tau_i = (C_i, T_i)$  and  $\tau_j = (C_j, T_j)$  we have  $\tau_i \prec \tau_j \implies T_i \leq T_j$ , then the operations must be scheduled in an increasing order of their periods corresponding to classical fixed priorities [10, 6]. We re-index operations in such a way that  $\tau_1 \prec \tau_2 \prec \dots \prec \tau_n$ , that is to say  $\tau_1$  precedes  $\tau_2$ ,  $\tau_2$  precedes  $\tau_3$  and so on. In the context of this report we shall use the term “level” rather than priority, level 1 which corresponds to operation  $\tau_1$  being the highest, and level  $n$  which corresponds to operation  $\tau_n$  being the lowest.

The **strict periodicity** constraint means that the start times  $s_i^k$  and  $s_i^{k+1}$  of two consecutive instances corresponding to operation  $\tau_i$  are **exactly** separated by its period:  $s_i^{k+1} - s_i^k = T_i, \forall k \geq 0$ . The instance started at time  $s_i^0 + kT_i$  has  $s_i^0 + (k+1)T_i$  as its deadline, i.e. the start time of the next instance.

For such a system of operations with precedence and strict periodicity constraints, we propose a method to compute on the one hand the exact number of preemptions, and on the other hand the schedule of the system when no idle time is allowed, i.e. the processor will always execute an operation as soon as it is possible to do so. Although idle time may help the system to be schedulable, when no idle time is allowed it is easier to find the start times of all the instances of an operation according to the precedence relation.

For the sake of readability and without any loss of generality, from now on, although it is not entirely realistic, we will consider the cost of one preemption for the processor to be  $\alpha = 1$  time unit in all the examples. This high cost of preemptions in terms of the execution time of operations is used to illustrate the impact of not counting the preemptions correctly. In addition, it is worth noticing that the analysis performed here would work even if the preemption cost were not a constant.

The remainder of the report is structured as follows: section 2 describes the model and gives the notations used throughout this report. Section 3 provides the definitions we need to take into account the exact number of preemptions in the schedulability analysis presented in section 4. That section explains in detail, on the one hand, our scheduling algorithm which counts the exact number of preemptions and, on the other hand, derives the new schedulability condition. The complexity of our algorithm is discussed in section 5. We conclude and propose future work in section 6.

## 2 Model

The model depicted in figure 1 is an extension, with preemption, of our previous model [1] for systems with precedence and strict periodicity constraints executed on a single processor.

Throughout the report, we assume that all timing characteristics are non negative integers, i.e. they are multiples of some elementary time interval (for example the “CPU tick”, the smallest indivisible CPU time unit):

$\tau_i = (C_i, T_i)$ : an operation,

$T_i$ : Period of  $\tau_i$ ,

$C_i$ : WCET of  $\tau_i$  without any preemption approximation,  $C_i \leq T_i$ ,

$\alpha$ : Temporal cost of one preemption for a given processor,

$\tau_i^k$ : The  $k^{th}$  instance of  $\tau_i$ ,

$N_p(\tau_i^k)$ : Exact number of preemptions of  $\tau_i$  in  $\tau_i^k$ ,

$C_i^k = C_i + N_p(\tau_i^k) \cdot \alpha$ : Preempted Execution Time (PET) of  $\tau_i$  including its exact preemption cost in  $\tau_i^k$ ,

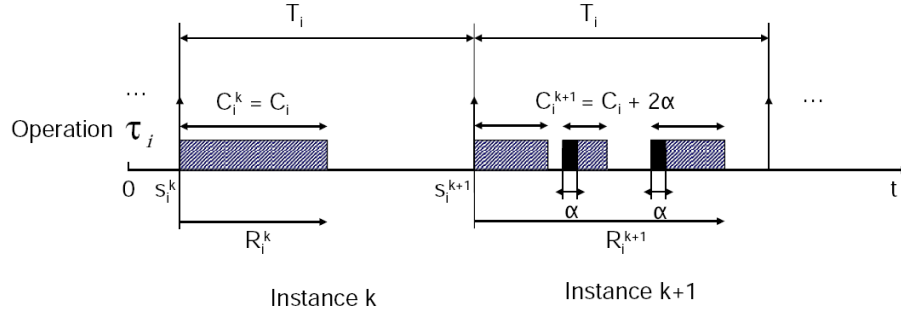


Figure 1: Model

$s_i^0$ : Start time of the first instance of  $\tau_i$ ,  
 $s_i^k = s_i^0 + (k-1)T_i$ : Start time of  $\tau_i^k$ ,  
 $R_i^k$ : Response time of  $\tau_i^k$ ,  
 $R_i$ : Worst-case response time of  $\tau_i$ .

In order to be consistent with the previous section, given a set of  $n$  operations, a *valid schedule*  $S$  for the system taking into account the exact number of preemptions will be yielded by the set of the start times of the first instance for all operations:

$$S = \{(s_1^0, s_2^0, \dots, s_n^0)\} \quad (1)$$

Indeed, the start time of the  $k^{th}$  instance of any operation  $\tau_i$  is derived from the start time  $s_i^0$  of the first instance thanks to the strict periodicity constraint:  $s_i^k = s_i^0 + (k-1)T_i$ .

It is worth noticing that since all the operations except the one with the shortest period w.r.t. the precedence relations may be preempted, the execution time of an operation may vary from one instance to another due to the number of preemptions. Therefore, the *preempted execution time* (PET) [13] which corresponds to the WCET augmented with the exact cost due to preemptions for each instance of an operation may also vary from one instance to another. Consequently, the PET denoted  $C_i^k$  for instance  $\tau_i^k$  of operation  $\tau_i$  depends on the instance and on the number of preemptions occurring in that instance. Its computation will be detailed below.

Because we intend to take into account the exact number of preemptions, and because all operations may be preempted, except the first one, i.e. the one with the shortest period, all instances of all operations must be considered since the number of preemptions may be different from one instance to another. We give a schedulability condition for each operation individually according to operations with shorter periods. For each operation, our scheduling algorithm first provides the start time of the first instance, then computes the exact number of preemptions per instance. This individual operation analysis leads, at the end, to a schedulability condition for all operations.

It has been shown in [1, 6, 10] that systems with precedence and strict periodicity constraints repeat identically after a time called the *hyperperiod* which corresponds to the Least Common Multiple (LCM) of the periods of all the operations.



### 3 Definitions

All the definitions and terminologies used in this section are directly inspired by [13] and are applied here to the case of a model with precedence and strict periodicity constraints.

From the point of view of any operation  $\tau_i$ , we define the *hyperperiod at level  $i$* ,  $H_i$ , which is given by  $H_i = LCM\{T_j\}_{\tau_j \in sp(\tau_i)}$ , where  $sp(\tau_i)$  is the set of operations with a period shorter than that of operation  $\tau_i$ . It is obvious that  $H_i$  time units after the first start time  $s_i^0$  of operation  $\tau_i$ , the start time of the next instance is exactly the same as that of  $s_i^0$  w.r.t. the start time of the first instances of operations preceding  $\tau_i$ . This characteristic derives from both the precedence and the strict periodicity constraints. Without any loss of generality we assume that the first operation  $\tau_1$  starts its execution at time  $t = 0$ . Since at each level  $i$  the schedule of  $\tau_i$  repeats indefinitely, it is sufficient to perform the scheduling analysis in the interval  $[s_i^0, s_i^0 + H_i]$  for  $\tau_i$  and  $[0, s_n^0 + H_n]$  for the whole set of operations. Therefore,  $\tau_i$  starts  $\sigma_i$  times in each hyperperiod at level  $i$  starting from 0, with

$$\sigma_i = \frac{H_i}{T_i} = \frac{LCM\{T_j\}_{\tau_j \in sp(\tau_i)}}{T_i} \quad (2)$$

Because operation  $\tau_i$  may only be preempted by the set of operations with a period shorter than  $\tau_i$  denoted  $sp(\tau_i)$ , then there are exactly  $\sigma_i$  different PETs for operation  $\tau_i$ . In other words, from the point of view of any operation  $\tau_i$ , we can define the function  $\pi : \mathbb{N}^+ \times \mathbb{N}^+ \rightarrow \mathbb{N}^{+\sigma_i} \times \mathbb{N}^+$ , where  $\pi(C_i, T_i) = \pi(\tau_i) = ((C_i^1, C_i^2, \dots, C_i^{\sigma_i}), T_i)$ , which maps the WCET  $C_i$  of operation  $\tau_i$  into its respective PET  $C_i^k$  in each instance  $\tau_i^k$  when  $\tau_i$  is schedulable. Consequently, we define the *exact total utilization factor* to be

$$U_n^* = \sum_{i=1}^n \frac{1}{\sigma_i} \left( \sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_n + \sum_{i=1}^n \frac{1}{\sigma_i} \left( \sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \quad (3)$$

where  $U_n = \sum_{i=1}^n \frac{C_i}{T_i}$ . Therefore, the exact cost due to preemptions incurred by the system is

$$\varepsilon_n = \sum_{i=1}^n \frac{1}{\sigma_i} \left( \sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \quad (4)$$

For a given set of  $n$  operations, we define the *exact total utilization factor at level  $j$* ,  $1 \leq j \leq n$  to be

$$U_j^* = \sum_{i=1}^j \frac{1}{\sigma_i} \left( \sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_j + \sum_{i=1}^j \frac{1}{\sigma_i} \left( \sum_{k=1}^{\sigma_i} \frac{N_p(\tau_i^k) \cdot \alpha}{T_i} \right) \quad (5)$$

Because of the precedence constraints among operations and because we proceed the schedule from the operation with the shortest period towards the operation with the longest period. At each level of the scheduling process the goal is to fill available time units in the previous schedule thus far obtained, with slices of the WCET of the current operation taking into account the exact number of preemptions, and hence we obtain the next current schedule. Consequently, we represent the previous schedule of every instance  $\tau_i^k$  of the current operation  $\tau_i = (C_i, T_i)$  by an ordered set of  $T_i$  time units where some are already executed because of the execution of operations with shorter periods

relatively to  $\prec$ , and the others are still available for the execution of operation  $\tau_i$  in that instance. We call this ordered set which describes the state of each instance  $\tau_i^k$  the  $\mathcal{M}_i^k$   $T_i$ -mesoid. We denote a time unit already executed by an “e” and a time unit still available by an “a”. The switch from an  $a$  to an  $e$  represents a preemption if the WCET of the current operation is strictly greater than the cardinal of the sub-set corresponding to the first sequence of  $a$ . Depending on the remaining execution time while filling available time units, this situation may occur again leading therefore to several preemptions which themselves may result in causing others. The cardinal of a sub-set corresponding to a sequence of consecutive time units already executed is called a *consumption*. It will be denoted by its value inside brackets. We enumerate the sequence of available time units according to natural numbers. This enumeration is done from the end of the first sequence of time units already executed in that instance. Each of these natural numbers corresponds to the number of available time units since the end of the first consumption. They represent all the possible PETs of the operation under consideration in the corresponding instance. Each of these natural numbers  $a_i$  is called an *availability*. For example, the 13-mesoid  $\{e, e, e, a, a, a, e, e, a, a, e, a, a\}$  will be represented by  $\{(3), 1, 2, 3, (2), 4, 5, (1), 6, 7\}$ ,  $(3), (2), (1)$  are consumptions and  $1, 2, 3, 4, 5, 6, 7$  are availabilities. More details on the definition of a  $T_i$ -mesoid are given in [13].

From the point of view of the current operation  $\tau_i = (C_i, T_i)$ , there are as many  $T_i$ -mesoids as instances in the hyperperiod  $H_i$  at level  $i$ , because operation  $\tau_i$  may only be preempted by operations in  $sp(\tau_i)$ . Therefore, there are  $\sigma_i$   $T_i$ -mesoids in  $H_i$  which will form a sequence of  $T_i$ -mesoids. We call  $\mathcal{L}_i^b = \{\mathcal{M}_i^{b,1}, \mathcal{M}_i^{b,2}, \dots, \mathcal{M}_i^{b,\sigma_i}\}$  the sequence of  $\sigma_i$   $T_i$ -mesoids **before**  $\tau_i$  is scheduled in the current schedule. The process used to build the sequence  $\mathcal{L}_i^b$  of operation  $\tau_i$  will be detailed later.

Still from the point of view of operation  $\tau_i$ , we define for each mesoid  $\mathcal{M}_i^{b,k}$ ,  $1 \leq k \leq \sigma_i$  of sequence  $\mathcal{L}_i^b$  the corresponding *universe*  $X_i^k$  to be the ordered set, compatible with that of the corresponding mesoid, which consists of all the availabilities of  $\mathcal{M}_i^{b,k}$ . That is to say, all the possible values that  $C_i^k$  can take in  $\mathcal{M}_i^{b,k}$ . Recall that  $C_i^k$  denotes the PET of  $\tau_i$  in  $\tau_i^k$ , the  $k^{th}$  instance of  $\tau_i$ .

Operation  $\tau_i$  will be said to be *potentially schedulable* if and only if

$$\begin{cases} C_i \in X_i^k & \forall k \in \{1, \dots, \sigma_i\} \\ \mathcal{M}_i^{b,k} \text{ starts with an available time unit} \\ \text{for each } k \in \{1, \dots, \sigma_i\} \end{cases} \quad (6)$$

The first  $\sigma_i$  equations of (6) verify that  $C_i$  belongs to each universe at level  $i$ . Then, the next  $\sigma_i$  equations verify that every  $\mathcal{M}_i^{b,k}$  starts with an availability as no idle time is allowed. These verifications are necessary for the strict periodicity constraints to be satisfied. As a matter of fact, if a  $T_i$ -mesoid starts with a consumption it is not possible to fill the previous schedule with slices of the WCET of the current operation  $\tau_i$  taking into account the cost of preemption as it belongs to a lower level than those already scheduled w.r.t  $\prec$ . Therefore its start time is postponed to the end of the consumption in the previous schedule, and thus does not satisfy the strict periodicity constraint of  $\tau_i$ . In this case the system is not schedulable. Notice that when this situation arises the three non schedulability conditions given in [9] hold.

Since  $C_i \in \{1, 2, \dots, T_i\}$ ,  $\forall 1 \leq i \leq n$ , let us define the following binary relation on each instance.

$\mathcal{R}$ : “availability  $a_{i_1}$  leads to the same number of preemptions as availability  $a_{i_2}$ ”,  
 $a_{i_1}, a_{i_2} \in \{1, 2, \dots, T_i\}$

$\mathcal{R}$  is clearly an equivalence relation on  $\{1, 2, \dots, T_i\}$  (reflexive, symmetric, transitive). Now, since  $X_i^k \subseteq \{1, 2, \dots, T_i\}$ ,  $\forall 1 \leq k \leq \sigma_i$ , thus  $\mathcal{R}$  is also an equivalence relation on  $X_i^k$ ,  $\forall 1 \leq k \leq \sigma_i$  and each  $X_i^k, k = 1, \dots, \sigma_i$  together with  $\mathcal{R}$  is a *setoid*<sup>3</sup>. From now on, we consider only the restriction of  $\mathcal{R}$  on  $X_i^k, k = 1, \dots, \sigma_i$  because  $X_i^k$  represents all the available time units in instance  $\tau_i^k$ .

Each  $T_i$ -mesoid consists of a sequence of time units already executed, i.e. consumptions, due to the schedule of operations with shorter periods, followed or preceded by a sequence of times units still available, i.e. availabilities. Actually, if  $C_i$  fits in the first sequence of consecutive availabilities in a  $T_i$ -mesoid, then no preemption occurs. Since each switch from an available time unit to an already executed time unit possibly corresponds to a preemption, then according to the value of  $C_i$  several preemptions may occur. Among the possible values that  $C_i$  can take, those which will lead to the same number of preemptions will be said to be equivalent w.r.t. to  $\mathcal{R}$ , and thus will belong to the same *equivalence class*. Therefore, the *equivalence classes* of each universe correspond to the subsets of availabilities determined by two consecutive consumptions in the associated mesoid.

Since we proceed the schedule from the operation with the shortest period to the operation with the longest period w.r.t. the precedence relations, it is obvious that the start time of the first instance  $s_i^0$  of operation  $\tau_i$  occurs at least after the end time of that of operation  $\tau_{i-1}$  in order to satisfy the strict periodicity constraint. Moreover,  $s_i^0$  occurs as soon as possible since no idle time is allowed. The latter statement implies that operation  $\tau_i$  starts  $\Delta_{i-1}$  time units after the start time  $s_{i-1}^0$  of the first instance of operation  $\tau_{i-1}$ . The computation of  $\Delta_{i-1}$  will be detailed later on. Already, it is worth noticing that  $\Delta_{i-1}$  is longer than or equal to the response time of  $\tau_{i-1}$  in its first instance because when the last piece of the PET of  $\tau_{i-1}$  fits exactly a sequence of consecutive availabilities, then the start time of the first instance of  $\tau_i$  is postponed. Hence, we can derive the first start time of any potentially schedulable operation  $\tau_i$  as the **sum** of the start time  $s_{i-1}^0$  of the first instance of operation  $\tau_{i-1}$  and  $\Delta_{i-1}$ .

$$s_i^0 = s_{i-1}^0 + \Delta_{i-1} \quad \forall i \in \{2, 3, \dots, n\} \quad (7)$$

When equation (6) holds for a given operation  $\tau_i$ , we call

$$\mathcal{L}_i^a = \left\{ \mathcal{M}_i^{a,1}, \mathcal{M}_i^{a,2}, \dots, \mathcal{M}_i^{a,\sigma_i} \right\}$$

the sequence of  $\sigma_i$   $T_i$ -mesoids of operation  $\tau_i$  **after**  $\tau_i$  is scheduled.  $\mathcal{L}_i^a$  is a function of  $\mathcal{L}_i^b$  which itself is a function of  $\mathcal{L}_{i-1}^a$ , both detailed as follows.

Thanks to everything we have presented up to now, we can assume without any loss of generality that the start time of the first instance of the operation with the shortest period, here  $\tau_1$ , starts its execution at time  $t = 0$ , i.e.  $s_1^0 = 0$ .

Let  $f$  be the function such that  $\mathcal{L}_i^b = f(\mathcal{L}_{i-1}^a)$  which transforms the sequence  $\mathcal{L}_{i-1}^a$  of  $\sigma_{i-1}$   $T_{i-1}$ -mesoids after operation  $\tau_{i-1}$  has been scheduled at level  $i - 1$  into the sequence  $\mathcal{L}_i^b$  of  $\sigma_i$   $T_i$ -mesoids before operation  $\tau_i$  is scheduled at level  $i$ .

As mentioned above, a mesoid consists only of time units already executed denoted by “e” and time units still available denoted by “a”. Moreover, the cardinal of a mesoid is equal to the period of the operation under consideration whatever the level is. As

<sup>3</sup>A setoid is a set equipped with an equivalence relation.

such, the function  $f$  transforms a time unit already executed (resp. still available) in the sequence  $\mathcal{L}_{i-1}^a$  into a time unit already executed (resp. still available) in the sequence  $\mathcal{L}_i^b$  by following an index  $\psi$  which enumerates according to natural numbers, the time units (already executed or still available) in the sequence  $\mathcal{L}_{i-1}^a$  of operation  $\tau_{i-1}$  after  $\tau_{i-1}$  is scheduled.  $\psi$  starts from the first available time unit of the first mesoid  $\mathcal{M}_{i-1}^{a,1}$  towards the last time unit of the last mesoid  $\mathcal{M}_{i-1}^{a,\sigma_{i-1}}$ , and then circles around to the beginning of the first mesoid  $\mathcal{M}_{i-1}^{a,1}$  again, until we get the  $\sigma_i$   $T_i$ -mesoids of  $\mathcal{L}_i^b$ . During this process each time  $\psi = T_i$ , a  $T_i$ -mesoid is obtained for the sequence  $\mathcal{L}_i^b$  and then the next  $T_i$ -mesoid is obtained by starting to count again from the next time unit to the current one. Indeed, the previous schedule at level  $i$  (the schedule obtained at level  $i-1$ ) consists of  $H_{i-1}$  time units whereas the schedule of the current operation  $\tau_i$  is computed upon  $H_i$  time units after the start time of its first instance  $s_i^0$ . That amounts to extending the previous schedule from  $H_{i-1}$  to  $H_i$  time units by identically repeating the previous schedule as often as necessary to obtain  $H_i$  time units.

Due to the precedence and strict periodicity constraints, notice that  $\psi$  in contrast to index  $\zeta$  used in [13] which started from the first time unit, starts from the first available time unit of the first  $T_{i-1}$ -mesoid as no idle time is allowed. The value of  $\Delta_{i-1}$  is therefore the consumption before the first available time unit in the sequence  $\mathcal{L}_{i-1}^a$  of operation  $\tau_{i-1}$ .

Since  $\tau_1$  is the operation with the shortest period,  $sp(\tau_1) = \{\tau_1\}$  and thus  $\sigma_1 = \frac{H_1}{T_1} = 1$  thanks to equation (2). Moreover, because  $\tau_1$  is never preempted, we have  $\mathcal{L}_1^b = \{\mathcal{M}_1^{b,1}\} = \{\{1, 2, \dots, T_1\}\}$  and  $\mathcal{L}_1^a = \{\mathcal{M}_1^{a,1}\} = \{\{(C_1), 1, 2, \dots, T_1 - C_1\}\}$ .

Let  $g$  be the function such that  $\mathcal{L}_i^a = g(\mathcal{L}_i^b)$  which transforms the sequence  $\mathcal{L}_i^b$  of  $\sigma_i$   $T_i$ -mesoids before operation  $\tau_i$  has been scheduled at level  $i$  into the sequence  $\mathcal{L}_i^a$  of  $\sigma_i$   $T_i$ -mesoids after operation  $\tau_i$  has been scheduled at level  $i$ .

For each  $T_i$ -mesoid  $\mathcal{M}_i^{b,k}$ ,  $k = 1, \dots, \sigma_i$  of  $\mathcal{L}_i^b$  we compute the PET  $C_i^k$  that we add to all the consumptions appearing in that  $T_i$ -mesoid before the availability corresponding to that PET. This yields the response time  $R_i^k$  of operation  $\tau_i$  in instance  $\tau_i^k$ . The PET, which takes into account the exact cost of preemption, is computed by using a fixed-point algorithm which is detailed in the next section. Now, for each  $T_i$ -mesoid of  $\mathcal{L}_i^b$ , function  $g$  transforms a time unit already executed in the sequence  $\mathcal{L}_i^b$  into a time unit already executed in the sequence  $\mathcal{L}_i^a$ , and transforms a time unit still available into either a time unit still available or a time unit already executed w.r.t. the following condition. We use an index which enumerates using numerals the time units from the first to the last one in each  $T_i$ -mesoid  $\mathcal{M}_i^{b,k}$  of  $\mathcal{L}_i^b$ . If the current value of the index is less than or equal to  $R_i^k$ , then function  $g$  transforms the time unit still available into a time unit already executed due to the execution of instance  $\tau_i^k$ , otherwise  $g$  transforms it into a time unit still available. Indeed, function  $g$  fills available time units in the current schedule with slices of the PETs in each  $T_i$ -mesoid, leading to the previous schedule for the next operation at level  $i+1$  w.r.t  $\prec$ .

To summarize, for every task  $\tau_i$ , we have

$$\tau_i : \begin{cases} \mathcal{L}_i^b = \{\mathcal{M}_i^{b,1}, \mathcal{M}_i^{b,2}, \dots, \mathcal{M}_i^{b,\sigma_i}\} \\ \mathcal{L}_i^a = \{\mathcal{M}_i^{a,1}, \mathcal{M}_i^{a,2}, \dots, \mathcal{M}_i^{a,\sigma_i}\} \end{cases}$$

where the start time of its first instance  $s_i^0$  is given by equation (7).

## 4 The proposed approach

In this section, before presenting our scheduling algorithm, we first outline our approach for a system of two operations and then for an arbitrary number of operations. This approach leads to a new schedulability condition for hard real-time systems with precedence and strict periodicity constraints using the exact number of preemptions and no idle time allowed. This condition is new in the sense that it takes into account the exact number of preemptions in the schedulability analysis for such systems rather than using an approximation in the WCETs.

Since the schedule proceeds from the operation with the shortest period corresponding to the highest level, to the one with the longest period corresponding to the lowest level, then for every potentially schedulable operation, we determine its schedule thanks to those with shorter periods.

In this process at each level  $i$  the basic idea consists in filling availabilities in each mesoid of the sequence  $\mathcal{L}_i^b$ , before operation  $\tau_i$  is scheduled, with slices (cardinal of equivalence classes) of its inflated WCET while taking into account the cost of the exact number of preemptions necessary for its schedule. At each preemption occurrence,  $\alpha$  time units add to the remaining execution time of the instance of the operation under consideration. This situation may occur again w.r.t. the remaining execution time, leading therefore to several preemptions which themselves may cause others. This is why it is crucial to calculate the exact number of preemptions. Finally, we obtain for each mesoid the PET, and then the corresponding response time. Determining the worst case among these response times allow us to conclude on the schedulability of operation  $\tau_i$  w.r.t.  $\prec$ . When  $\tau_i$  is schedulable, we build the sequence  $\mathcal{L}_i^a$ , after  $\tau_i$  is scheduled, in order to check the schedulability of the next operation, and so on, otherwise the system is not schedulable.

### 4.1 Scheduling of two operations

Let us justify the general result of our approach by considering the simple case of the scheduling problem of two operations  $\tau_1 = (C_1, T_1)$  and  $\tau_2 = (C_2, T_2)$ , with  $\tau_1 \prec \tau_2$ . Thanks to everything we have presented up to now,  $\tau_1$  is scheduled first,  $s_1^0 = 0$ , and  $T_1 \leq T_2$ . The latter statement implies that **before**  $\tau_1$  is scheduled, its PET can potentially take any value from 1 up to the value of its period  $T_1$ . Since operation  $\tau_1$  is never preempted, then  $\sigma_1 = 1$  and  $C_1^k = C_1, \forall k \geq 1$  and  $\tau_1' = \pi(\tau_1) = ((C_1), T_1)$ . Therefore,  $\mathcal{L}_1^b = \{\mathcal{M}_1^{b,1}\} = \{\{1, 2, \dots, T_1\}\}$  and  $X_1^1 = \{1, 2, \dots, T_1\}$ . In addition, its response time is also equal to  $C_1$ . Hence, **after** being scheduled,  $\tau_1$  has consumed  $C_1$  time units, and thus there remain  $T_1 - C_1$  availabilities in each of its instances. Consequently, the corresponding  $T_1$ -mesoids associated to operation  $\tau_1$  are given by

$$\tau_1 : \begin{cases} \mathcal{L}_1^b = \{\mathcal{M}_1^{b,1}\} = \{\{1, 2, \dots, T_1\}\} \\ \mathcal{L}_1^a = \{\mathcal{M}_1^{a,1}\} = \{\{(C_1), 1, 2, \dots, T_1 - C_1\}\} \end{cases}$$

Now, we have to schedule task  $\tau_2$  by taking into account the exact number of preemptions. Thanks to the previous section we have:  $\Delta_1 = C_1$ ,  $s_2^0 = s_1^0 + \Delta_1 = C_1$ , and  $\mathcal{L}_2^b = f(\mathcal{L}_1^a)$  the sequence of  $\sigma_2 = \frac{H_2}{T_2}$   $T_2$ -mesoids. Then, we can easily determine the universe  $X_2^k$  corresponding to each  $T_2$ -mesoid  $\mathcal{M}_2^{b,k}$ ,  $k = 1, \dots, \sigma_2$ . Thus, thanks to

equation (6), operation  $\tau_2$  is *potentially schedulable* if and only if

$$\begin{cases} C_2 \in X_2^k \quad \forall k \in \{1, \dots, \sigma_2\} \\ \mathcal{M}_2^{b,k} \text{ starts with an available time unit} \\ \text{for each } k \in \{1, \dots, \sigma_2\} \end{cases} \quad (8)$$

We give the following example in order to illustrate these conditions. Let us consider a set of two operations  $\tau_1 = (2, 6)$  and  $\tau_2 = (4, 9)$ . We have  $s_1^0 = 0$  and

$$\tau_1 : \begin{cases} \mathcal{L}_1^b = \{\mathcal{M}_1^{b,1}\} = \{\{1, 2, 3, 4, 5, 6\}\} \\ \mathcal{L}_1^a = \{\mathcal{M}_1^{a,1}\} = \{\{(2), 1, 2, 3, 4\}\} \end{cases}$$

Since  $\Delta_1$  equals the first consumption, here (2), then we have  $\Delta_1 = 2$ ,  $s_2^0 = s_1^0 + \Delta_1 = 0 + 2 = 2$  and  $\sigma_2 = \frac{H_2}{T_2} = \frac{LCM(6, 9)}{9} = 2$ , we thus derive  $\mathcal{L}_2^b = f(\mathcal{L}_1^a)$  which consists of a sequence of two 9-mesoids. We obtain

$$\begin{aligned} \mathcal{L}_2^b &= \{\mathcal{M}_2^{b,1}, \mathcal{M}_2^{b,2}\} \\ &= \{\{1, 2, 3, 4, (2), 5, 6, 7\}, \{1, (2), 2, 3, 4, 5, (2)\}\} \end{aligned}$$

For each 9-mesoid  $\mathcal{M}_2^{b,k}$ ,  $1 \leq k \leq 2$ , composing  $\mathcal{L}_2^b$ , we build the corresponding universe  $X_2^k$ ,  $1 \leq k \leq 2$ . These universes are given by

$$\tau_2 : \begin{cases} X_2^1 = \{1, 2, 3, 4, 5, 6, 7\} \\ X_2^2 = \{1, 2, 3, 4, 5\} \end{cases}$$

From these universes, we deduce that operation  $\tau_2$  is potentially schedulable because equation (6) is satisfied. Indeed, for each resulting universe  $X_2^k$ , we have

$$\begin{cases} 4 \in X_2^1 \text{ and } 4 \in X_2^2 \\ \mathcal{M}_2^{b,1} \text{ and } \mathcal{M}_2^{b,2} \text{ start with an available time unit} \end{cases}$$

Now, thanks to the equivalence relation  $\mathcal{R}$  on each universe  $X_2^k$  for  $k = 1, 2$ , the *equivalence classes* are given by

for universe  $X_2^1$ :  $[0]^1 = \{1, 2, 3, 4\}$  and  $[1]^1 = \{5, 6, 7\}$

for universe  $X_2^2$ :  $[0]^2 = \{1\}$  and  $[1]^2 = \{2, 3, 4, 5\}$

where for  $m \in \mathbb{N}$  and  $1 \leq k \leq \sigma_2$ ,  $[m]^k$  denotes the subset of  $X_2^k$  composed of the availabilities which are preempted  $m$  times. Thus, for this example,  $\mathcal{L}_2^b = f(\mathcal{L}_1^a)$  can also be written by displaying the equivalence classes as follows

$$\mathcal{L}_2^b = \{\{\overbrace{1, 2, 3, 4}^{[0]^1}, \overbrace{(2), 5, 6, 7}^{[1]^1}\}, \{\overbrace{1}^{[0]^2}, \overbrace{(2), 2, 3, 4, 5, (2)}^{[1]^2}\}\}$$

Here we have all we need to compute the exact number of preemptions  $N_p(\tau_2^k)$  and then the corresponding PET  $C_2^k$  of operation  $\tau_2$  in its  $k^{th}$  instance,  $1 \leq k \leq 2$ .

Since operation  $\tau_2$  is potentially schedulable (equation (8) holds), its WCET  $C_2$  belongs to one of the possible *equivalence classes* in the universe  $X_2^k$ ,  $1 \leq k \leq \sigma_2$  (see figure 3). We call  $[\theta_1]^k$  that equivalence class. Because we take into account the exact number of preemptions the PET  $C_2^k = C_2 + N_p(\tau_2^k) \cdot \alpha$  will actually belong to the equivalence class  $[\theta_m]^k$  with  $m \geq 1$ . It is worth noticing that if  $[\theta_1]^k = [0]^k$  then  $C_2^k = C_2$ .

In order to determine the actual equivalence class  $[\theta_m]^k$  we need to compute the exact number of preemptions  $N_p(\tau_2^k)$ . This is achieved by adding the number of preemptions  $\theta_1$  incurred by the WCET  $C_2$  and the number of preemptions due to the preemptions themselves.

In each universe  $X_2^k$ ,  $1 \leq k \leq \sigma_2$ , the number of preemptions  $N_p(\tau_2^k)$  and the PET  $C_2^k$  of operation  $\tau_2$  are computed by using the following iterative algorithm.

$$\begin{cases} \theta_0 = 0 \\ C_2^{k,0} = C_2 \in [\theta_1]^k \\ C_2^{k,m} = C_2^{k,m-1} + (\theta_m - \theta_{m-1}) \cdot \alpha \in [\theta_m]^k \quad \forall m \geq 1 \end{cases}$$

This means that the PET is computed from its initial value equal to the WCET  $C_2$  to which is iteratively added the time corresponding to the difference between the value of the equivalence class of the current PET and that of the previous one. The current PET is a step function of this difference. This computation stops as soon as either two consecutive values of  $C_2^{k,j}$ ,  $j \geq 1$  are equal, i.e. they belong to the same *equivalence class*  $[\theta_l]^k$ ,  $l \geq 0$  (see figure 4), or there exists  $\mu_1 \geq 1$  such that  $C_2^{k,\mu_1} > \text{card}(X_2^k)$ . In this latter case operation  $\tau_2$  is not schedulable because the deadline of the operation has been exceeded. In the first case we have

$$C_2^k = C_2 + \sum_{j=1}^l (\theta_j - \theta_{j-1}) \cdot \alpha = C_2 + N_p(\tau_2^k) \cdot \alpha \quad (9)$$

Consequently, the image of  $\tau_2$  by function  $\pi$  is given by

$$\tau_2' = \pi(\tau_2) = ((C_2^1, C_2^2, \dots, C_2^{\sigma_2}), T_i) \quad (10)$$

The response time  $R_2^k$ ,  $1 \leq k \leq \sigma_2$  of task  $\tau_2$  in its  $k^{\text{th}}$  instance, i.e. in the  $k^{\text{th}}$   $T_2$ -mesoid is obtained by summing  $C_2^k$  with all the consumptions appearing before  $C_2^k$  in the corresponding mesoid. Once this has been done, the worst-case response time  $R_2$  of task  $\tau_2$  is given by

$$R_2 = \max_{\{1 \leq k \leq \sigma_2\}} (R_2^k)$$

Thus the sequence  $\mathcal{L}_2^a = g(\mathcal{L}_2^b)$  can be built.

We still assume  $\alpha = 1$  to be the cost of one preemption for the processor in order to clearly show the impact of the preemption. In this example we recall that operation  $\tau_2 = (4, 9)$  is potentially schedulable.

In the first universe corresponding to instance  $\tau_2^1$ ,  $C_2 = 4 \in [0]^1$ ; thus  $C_2^1 = C_2 = 4$ , whereas in the second universe corresponding to instance  $\tau_2^2$ ,  $C_2 = 4 \in [1]^2$ . The computation of  $N_p(\tau_2^2)$  is obtained as follows.

$$\begin{aligned} \theta_0 &= 0 \\ C_2^{2,0} &= C_2 = 4 \in [1]^2 \rightarrow \theta_1 = 1 \\ C_2^{2,1} &= C_2^{2,0} + (\theta_1 - \theta_0) \cdot \alpha = 5 \in [1]^2 \rightarrow \theta_2 = 1 \\ C_2^{2,2} &= C_2^{2,1} + (\theta_2 - \theta_1) \cdot \alpha = 5 \in [1]^2 \rightarrow \theta_3 = 1 \end{aligned}$$



Since  $C_2^{2,1} = C_2^{2,2}$ , we get  $N_p(\tau_2^2) = \theta_3 = 1$  and thus we obtain  $C_2^2 = 4 + 1 \cdot 1 = 5$ . Hence, the image of operation  $\tau_2$  by function  $\pi$  is given by  $\tau_2' = \pi(\tau_2) = ((4, 5), 9)$ .

Thanks to our previous definition,  $C_2^1 = 4$ , and there is no consumption appearing before 4 as it belongs to  $[0]^1$ , thus the response time of operation  $\tau_2$  in the first mesoid is  $R_2^1 = 4 + 0 = 4$ .  $C_2^2 = 5$ , and there is one consumption appearing before 5 as it belongs to  $[1]^2$ , thus the response time of operation  $\tau_2$  in the second mesoid is  $R_2^2 = 5 + 2 = 7$ . Hence, the worst-case response time  $R_2$  of operation  $\tau_2$  is given by  $R_2 = 7 \leq T_2$ . Thus, operation  $\tau_2$  is schedulable, as is the system  $\{\tau_1, \tau_2\}$ .

Now we have everything to build  $\mathcal{L}_2^a = g(\mathcal{L}_2^b)$ . Indeed, from

$$\mathcal{L}_2^b = \{ \{ \overbrace{1, 2, 3, 4}^{[0]^1}, (2), \overbrace{5, 6, 7}^{[1]^1} \}, \{ \overbrace{1}^{[0]^2}, (2), \overbrace{2, 3, 4, 5}^{[1]^2}, (2) \} \}$$

since  $R_2^1 = 4$  and  $R_2^2 = 7$ , then the first four (resp. seven) time units in the first (resp. second) 9-mesoid have been executed, and consequently we have

$$\mathcal{L}_2^a = g(\mathcal{L}_2^b) = \{ \{(6), 1, 2, 3\}, \{(9)\} \}$$

Hence, by using expression (3), the exact total utilization factor of the processor is given by

$$U_2^* = \sum_{i=1}^2 \frac{1}{\sigma_i} \left( \sum_{k=1}^{\sigma_i} \frac{C_i^k}{T_i} \right) = U_2 + \frac{1}{\sigma_2} \left( \sum_{k=1}^{\sigma_2} \frac{N_p(\tau_2^k) \cdot \alpha}{T_2} \right) \quad (11)$$

Therefore,  $U_2^* = \frac{2}{6} + \frac{1}{2} \left( \frac{4+5}{9} \right) = 0.833$  whereas  $U_2 = \frac{2}{6} + \frac{4}{9} = 0.777$ , and thus the cost of preemption is  $\varepsilon_2 = 0.833 - 0.777 = 0.056$ .

Figure 2 depicts the schedule of this example taking into account the exact number of preemptions.

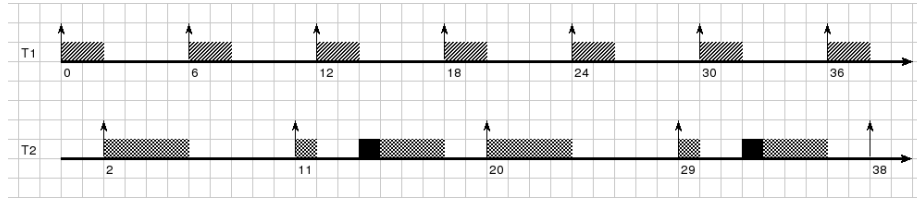


Figure 2: Execution of two operations with the exact number of preemptions

## 4.2 Scheduling of $n > 2$ tasks

The strategy that we will adopt in this section to compute both the exact number of preemptions and the PETs for a given operation in each of its instances, is the generalization of everything we presented in the previous section for the simple case of two operations.



### 4.3 Scheduling algorithm

We assume that the first  $i - 1$  operations with  $2 \leq i \leq n$  have already been scheduled, i.e. the sequence  $\mathcal{L}_{i-1}^a$  of operation  $\tau_{i-1}$  is known, and that we are about to schedule operation  $\tau_i$ , which is potentially schedulable, i.e.

$$\left\{ \begin{array}{l} C_i \in X_i^k \quad \forall k \in \{1, \dots, \sigma_i\} \\ \mathcal{M}_i^{b,k} \text{ starts with an available time unit} \\ \text{for each } k \in \{1, \dots, \sigma_i\} \end{array} \right.$$

As in the previous section for the construction of  $\mathcal{L}_2^b = f(\mathcal{L}_1^a)$ , sequence  $\mathcal{L}_i^b = f(\mathcal{L}_{i-1}^a)$  of operation  $\tau_i$  is built thanks to index  $\psi$  on sequence  $\mathcal{L}_{i-1}^a$  of operation  $\tau_{i-1}$  without forgetting to start at the first available time unit rather than the first time unit as in [13]. Again this is due to the constraints on the system and the fact that no idle time is allowed: the start time of the first instance of operation  $\tau_i$  is at  $s_i^0 = s_{i-1}^0 + \Delta_{i-1}$ . The sequence  $\mathcal{L}_i^b$  consists of  $\sigma_i$   $T_i$ -mesoids  $\mathcal{M}_i^{b,k}$  with  $k = 1, \dots, \sigma_i$  since operation  $\tau_i$  may only be preempted by operations belonging to  $sp(\tau_i)$ . We can therefore determine the universes  $X_i^k \quad \forall k \in \{1, \dots, \sigma_i\}$  when the sequence  $\mathcal{L}_{i-1}^a$  is known. The response time  $R_i^k$  of operation  $\tau_i$  in its  $k^{th}$  instance, i.e. in the  $k^{th}$   $T_i$ -mesoid will be obtained by summing  $C_i^k$  with all consumptions prior to  $C_i^k$  in the corresponding mesoid. The worst-case response time  $R_i$  of operation  $\tau_i$  will be given by

$$R_i = \max_{\{1 \leq k \leq \sigma_i\}} (R_i^k)$$

This equation leads us to say that operation  $\tau_i$  is schedulable if and only if

$$R_i \leq T_i \quad (12)$$

Again,  $\mathcal{L}_i^a = g(\mathcal{L}_i^b)$  will be deduced from sequence  $\mathcal{L}_i^b$  like  $\mathcal{L}_1^a = g(\mathcal{L}_1^b)$  in the previous section. For the sake of clarity, whenever there are two consecutive consumptions in the same *mesoid*, this amounts to considering only one consumption which is the sum of the previous consumptions. That is to say that after determining the response time of operation  $\tau_i$  in its  $k^{th}$  mesoid, if  $\mathcal{M}_i^{a,k} = \{(c_1), (c_2), 1, 2, \dots\}$ , then this is equivalent to  $\mathcal{M}_i^{a,k} = \{(c_1 + c_2), 1, 2, \dots\}$  without any loss of generality.

Below, we present our scheduling algorithm which, for a given operation, on the one hand first determines the start time of its first instance relatively to  $\prec$ , then counts the exact number of preemptions in each of its instances, and on the other hand provides its PET in each of its instances in order to take into account the exact number of preemptions in the schedulability condition. It has the following twelve steps. Since the operation with the shortest period, namely operation  $\tau_1$ , is never preempted, the loop starts from the index of the operation with the second shortest period, namely operation  $\tau_2$  as the schedule proceeds towards operations with longer periods.

- 1: **for**  $i = 2$  **to**  $n$  **do**
- 2:   Compute the number  $\sigma_i$  of times that operation  $\tau_i = (C_i, T_i)$  has started in the hyperperiod at level  $i$

$$\sigma_i = \frac{H_i}{T_i} = \frac{lcm\{T_j\}_{\tau_j \in sp(\tau_i)}}{T_i}$$

Recall that  $H_i = lcm\{T_1, T_2, \dots, T_i\}$

- 3: Determine the start time of the first instance of operation  $\tau_i$ :

$$s_i^0 = s_{i-1}^0 + \Delta_{i-1}$$

where  $\Delta_{i-1}$  is the consumption before the first available time unit in the sequence  $\mathcal{L}_{i-1}^a$  of operation  $\tau_{i-1}$ .

- 4: Build the sequence  $\mathcal{L}_i^b = f(\mathcal{L}_{i-1}^a)$  of  $T_i$ -mesoids of operation  $\tau_i$  before it is scheduled. This construction consists of  $\sigma_i$   $T_i$ -mesoids  $\mathcal{M}_i^{b,k}$  with  $k = 1, \dots, \sigma_i$ , and is based on a modulo  $T_i$  arithmetic using index  $\psi$  on the sequence  $\mathcal{L}_{i-1}^a$  without forgetting to start at the first available time unit rather than the first time unit as in [13]. This is due to the constraints and the fact that no idle time is allowed.
- 5: For each  $T_i$ -mesoid  $\mathcal{M}_i^{b,k}$  resulting from the previous step, build the corresponding universe  $X_i^k$  which consists of the ordered set of all availabilities of  $\mathcal{M}_i^{b,k}$ . Notice that this set corresponds to the set of all possible values that the PET  $C_i^k$  of operation  $\tau_i$  can take in  $\mathcal{M}_i^{b,k}$ .
- 6: Build all the equivalence classes for each universe  $X_i^k$ . An equivalence class of  $X_i^k$  is composed of the subset of availabilities determined by two consecutive consumptions in the associated mesoid  $\mathcal{M}_i^{b,k}$ .  $m \in \mathbb{N}$  in expression  $[m]^k$  denotes the subset of  $X_i^k$  composed of the availabilities which are preempted  $m$  times.
- 7: Compute both the exact number of preemptions and the PET  $C_i^k$  of operation  $\tau_i$  in each universe  $X_i^k$ ,  $1 \leq k \leq \sigma_i$ , resulting from the previous step thanks to the algorithm inlined in this step. Since  $\tau_i$  is potentially schedulable, i.e. its WCET  $C_i$  belongs to one and only one equivalence class  $[\theta_1]^k$  in each universe  $X_i^k$  (see figure 3), we must verify that it is actually schedulable given that some preemptions may occur whose costs actually add to the WCET.

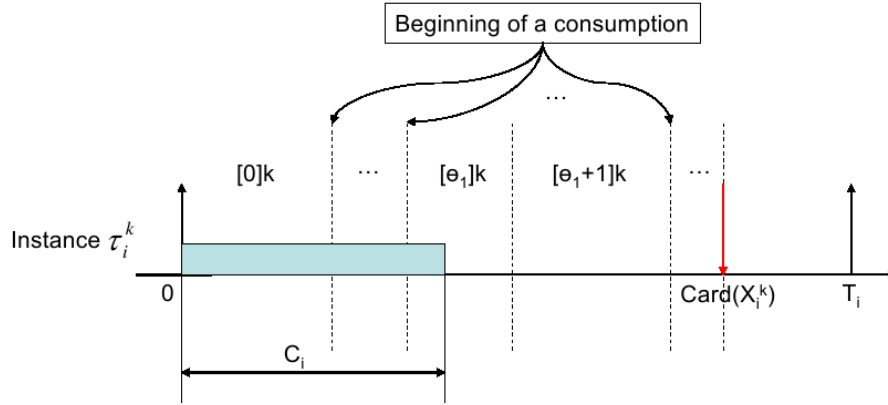


Figure 3: Operation  $\tau_i$  potentially schedulable

The recursive inflation of the execution time of the operation, due to preemptions, starts from the value of the WCET and stops when two consecutive values of the inflated WCET are equal, i.e. when the PET is reached. Indeed, the current inflated WCET is obtained by adding the previous inflated WCET and the

cost of preemptions incurred by this latter WCET. This explains the following fixed-point algorithm.

$$\begin{cases} \theta_0 = 0 \\ C_i^{k,0} = C_i \in [\theta_1]^k \\ C_i^{k,m} = C_i^{k,m-1} + (\theta_m - \theta_{m-1}) \cdot \alpha \in [\theta_m]^k \quad \forall m \geq 1 \end{cases}$$

This computation stops as soon as either two consecutive values of  $C_i^{k,j}$ ,  $j \geq 1$  are equal, i.e. they belong to the same equivalence class  $[\theta_l]^k$ ,  $l \geq 0$  (see figure 4) or there exists  $\mu_2 \geq 1$  such that  $C_i^{k,\mu_2} > \text{card}(X_i^k)$ . In the latter case, operation  $\tau_i$  is not schedulable because the deadline of the operation has been exceeded. In the first case we have

$$C_i^k = C_i + \sum_{j=1}^l (\theta_j - \theta_{j-1}) \cdot \alpha = C_i + N_p(\tau_i^k) \cdot \alpha \quad (13)$$

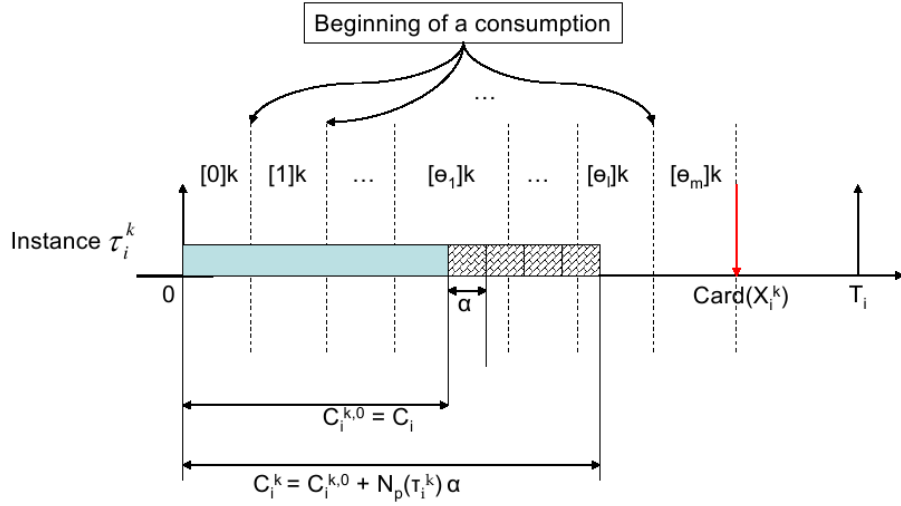


Figure 4: PET of operation  $\tau_i$  in instance  $\tau_i^k$ :  $C_i^k$

- 8: Deduce the image  $\tau_i' = \pi(\tau_i) = ((C_i^1, C_i^2, \dots, C_i^{\sigma_i}), T_i)$  of operation  $\tau_i$  resulting from the previous step.
- 9: Determine the response time  $R_i^k$ ,  $1 \leq k \leq \sigma_i$  of operation  $\tau_i$  in its  $k^{th}$  instance, i.e. in the  $k^{th}$   $T_i$ -mesoid. This is obtained by summing  $C_i^k$  with all the consumptions prior to  $C_i^k$  in the corresponding mesoid. Deduce the worst-case response time  $R_i$  of operation  $\tau_i$ .

$$R_i = \max_{\{1 \leq k \leq \sigma_i\}} (R_i^k)$$

It is worth noticing that operation  $\tau_i$  is schedulable if and only if

$$R_i \leq T_i.$$

- 10: If  $R_i \leq T_i$  then build the sequence  $\mathcal{L}_i^a = g(\mathcal{L}_i^b)$ , increment  $i$ , and go back to step 2 as long as there remain potentially schedulable operations in the system.
- 11: If  $R_i > T_i$ , then the system  $\{\tau_i = (C_i, T_i)\}_{1 \leq i \leq n}$  is not schedulable.
- 12: **end for**

Thanks to the above algorithm, a system of  $n$  operations  $\{\tau_i = (C_i, T_i)\}_{1 \leq i \leq n}$ , with precedence and strict periodicity constraints where no idle time is allowed and which takes into account the exact number of preemptions, is schedulable if and only if

$$R_i \leq T_i \quad \forall i \in \{1, 2, \dots, n\} \quad (14)$$

The exact total utilization factor of the processor is given by

$$U_n^* = \sum_{j=1}^n \frac{1}{\sigma_j} \left( \sum_{k=1}^{\sigma_j} \frac{C_j^k}{T_j} \right) = U_n + \sum_{j=1}^n \frac{1}{\sigma_j} \left( \sum_{k=1}^{\sigma_j} \frac{N_p(\tau_j^k) \cdot \alpha}{T_j} \right).$$

where  $U_n = \frac{C_i}{T_i}$ , and a valid schedule is given by

$$\mathcal{S} = \{(s_1^0, s_2^0, \dots, s_n^0)\}$$

### Example

Still with the same assumption that  $\alpha = 1$ , let us consider  $\{\tau_1, \tau_2, \tau_3, \tau_4\}$  to be a system of four operations with precedence and strict periodicity constraints and the characteristics defined in table 1.

Table 1: Characteristics of the operations

	$C_i$	$T_i$
$\tau_1$	4	10
$\tau_2$	4	15
$\tau_3$	2	20
$\tau_4$	7	60

According to the precedence constraints, the shorter the period of an operation is, the higher its level is. Thus, as depicted in table 1,  $\tau_1$  has the highest level and operation  $\tau_4$  the lowest level. Thanks to our scheduling algorithm,  $\sigma_1 = 1$ , thus for operation  $\tau_1$  whose first start time is  $s_1^0 = 0$ , we have

$$\tau_1 : \begin{cases} \mathcal{L}_1^b = \{\mathcal{M}_1^{b,1}\} = \{\{1, 2, 3, 4, 5, 6, 7, 8, 9, 10\}\} \\ \tau_1 = (4, 10) \mapsto \tau_1' = ((4), 10) \\ \mathcal{L}_1^a = \{\mathcal{M}_1^{a,1}\} = \{\{(4), 1, 2, 3, 4, 5, 6\}\} \end{cases}$$

$\sigma_2 = 2$  and  $\Delta_1 = 4$ , thus for operation  $\tau_2$  whose first start time is  $s_2^0 = s_1^0 + \Delta_1 = 4$ , we have

$$\tau_2 : \begin{cases} \mathcal{L}_2^b = f(\mathcal{L}_1^a) = \left\{ \mathcal{M}_2^{b,1}, \mathcal{M}_2^{b,2} \right\} \\ \quad = \left\{ \overbrace{\{1, 2, 3, 4, 5, 6, (4)\}}^{[0]^1}, \overbrace{\{7, 8, 9, 10, 11\}}^{[1]^1} \right\}, \\ \quad \quad \left\{ \overbrace{1}^{[0]^2}, (4), \overbrace{2, 3, 4, 5, 6, 7, (4)}^{[1]^2} \right\} \\ \tau_2 = (4, 15) \mapsto \tau_2' = ((4, 5), 15) \\ \mathcal{L}_2^a = g(\mathcal{L}_2^b) = \left\{ \mathcal{M}_2^{a,1}, \mathcal{M}_2^{a,2} \right\} \\ \quad = \left\{ \{(4), 1, 2, (4), 3, 4, 5, 6, 7\}, \{(9), 1, 2, (4)\} \right\} \end{cases}$$

$\sigma_3 = 3$  and  $\Delta_2 = 4$ , thus for operation  $\tau_3$  whose first start time is  $s_3^0 = s_2^0 + \Delta_2 = 8$ , we have

$$\tau_3 : \begin{cases} \mathcal{L}_3^b = f(\mathcal{L}_2^a) = \left\{ \mathcal{M}_3^{b,1}, \mathcal{M}_3^{b,2}, \mathcal{M}_3^{b,3} \right\} \\ \quad = \left\{ \overbrace{\{1, 2, (4)\}}^{[0]^1}, \overbrace{\{3, 4, 5, 6, 7, (9)\}}^{[1]^1} \right\}, \\ \quad \quad \left\{ \overbrace{1, 2}^{[0]^2}, (8), \overbrace{3, 4}^{[1]^2}, (4), \overbrace{5, 6, 7, 8}^{[2]^2} \right\}, \\ \quad \quad \left\{ \overbrace{1}^{[0]^3}, (9), \overbrace{2, 3}^{[1]^3}, (8) \right\} \\ \tau_3 = (2, 20) \mapsto \tau_3' = ((2, 2, 3), 20) \\ \mathcal{L}_3^a = g(\mathcal{L}_3^b) = \left\{ \mathcal{M}_3^{a,1}, \mathcal{M}_3^{a,2}, \mathcal{M}_3^{a,3} \right\} \\ \quad = \left\{ \{(6), 1, 2, 3, 4, 5, (9)\}, \{(10), 1, 2, (4), 3, 4, 5, 6\}, \right. \\ \quad \quad \left. \{(20)\} \right\} \end{cases}$$

$\sigma_4 = 1$  and  $\Delta_3 = 6$ , thus for operation  $\tau_4$  whose first start time is  $s_4^0 = s_3^0 + \Delta_3 = 14$ , we have

$$\tau_4 : \begin{cases} \mathcal{L}_4^b = f(\mathcal{L}_3^a) = \left\{ \mathcal{M}_4^{b,1} \right\} \\ \quad = \left\{ \overbrace{\{1, 2, 3, 4, 5, (19)\}}^{[0]^1}, \overbrace{\{6, 7, (4)\}}^{[1]^1}, \overbrace{\{8, 9, 10, 11, (26)\}}^{[2]^1} \right\} \\ \tau_4 = (7, 60) \mapsto \tau_4' = ((9), 60) \\ \mathcal{L}_4^a = g(\mathcal{L}_4^b) = \left\{ \mathcal{M}_4^{a,1} \right\} = \left\{ \{(32), 1, 2, (26)\} \right\} \end{cases}$$

Consequently, the set of operations  $\{\tau_1, \tau_2, \tau_3, \tau_4\}$  with precedence and strict periodicity constraints is schedulable and the valid schedule with no idle time allowed is given by  $\mathcal{S} = \{(s_1^0, s_2^0, s_3^0, s_4^0) = (0, 4, 8, 14)\}$ . Moreover

$$U_4^* = \frac{4}{10} + \frac{1}{2} \left( \frac{4+5}{15} \right) + \frac{1}{3} \left( \frac{2+2+3}{20} \right) + \frac{9}{60} = 0.966$$

whereas

$$U_4 = \frac{4}{10} + \frac{4}{15} + \frac{2}{20} + \frac{7}{60} = 0.883$$

Hence, the exact cost of preemption is  $\varepsilon_4 = 0.083$ . The schedule of this set of operations with precedence and strict periodicity constraints with the exact number of preemptions considered is depicted in figure 5,

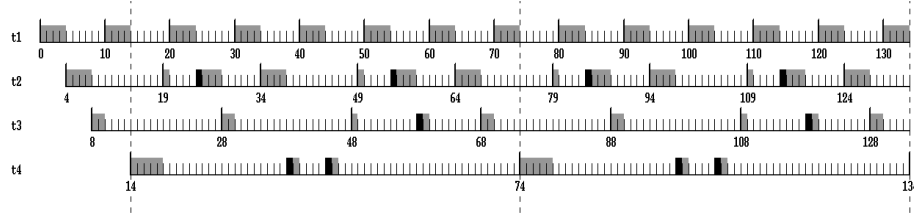


Figure 5: Execution of a set of operations considering the exact number of preemptions

whereas the schedule of the same set of operations with the cost of preemption approximated within WCETs is depicted in figure 6.

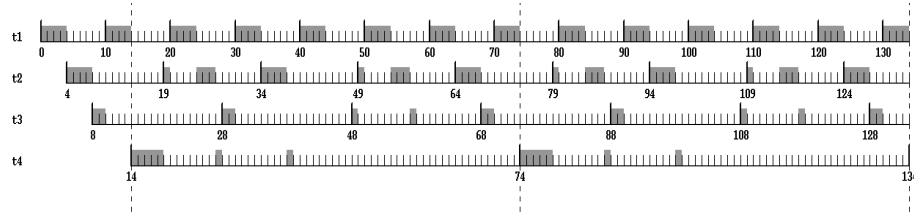


Figure 6: Execution of a set of operations considering the cost of preemptions approximated within WCETs

## 5 Complexity of the proposed scheduling algorithm

The complexity of the algorithm proposed here is similar to that proposed in [13].

## 6 Conclusion and future work

We are interested in hard real-time systems with precedence and strict periodicity constraints where it is mandatory to satisfy these constraints. We are also interested in preemption which offers great advantages when seeking schedules. Since classical approaches are based on an approximation of the cost of the preemption in WCETs, possibly leading to an incorrect real-time execution, we proposed an approach that takes its exact cost into account. We proposed a scheduling algorithm which counts the exact number of preemptions for a given system and thus gives a stronger schedulability condition than those in the literature.

Currently, we are adding the latency constraints to our model and we are planning to study the same problem when jitter is allowed on the periods of operations and

then, the complexity of our approach. Afterwards, because idle time may increase the possible schedules, we also plan to allow idle time, even though this would increase the complexity of the scheduling algorithm.

## References

- [1] L. Cucu, R. Kocik, and Y. Sorel. Real-time scheduling for systems with precedence, periodicity and latency constraints. In *Proceedings of 10th Real-Time Systems Conference, RTS'02*, Paris, France, March 2002.
- [2] Joseph Y.-T. Leung and M. L. Merrill. A note on preemptive scheduling of periodic, real-time tasks. *Information Processing Letters*, 1980.
- [3] A. Choquet-Geniet and E. Grolleau. Minimal schedulability interval for real time systems of periodic tasks with offsets. *Theoretical Computer Science*, 2003.
- [4] Ray Obenza and Geoff. Mendal. Guaranteeing real time performance using rma. *The Embedded Systems Conference, San Jose, CA*, 1998.
- [5] K. Ramamritham. Allocation and scheduling of precedence-related periodic tasks. *IEEE Transactions on Parallel and Distributed Systems*, 1995.
- [6] L. Cucu and Y. Sorel. Schedulability condition for systems with precedence and periodicity constraints without preemption. In *Proceedings of 11th Real-Time Systems Conference, RTS'03*, Paris, March 2003.
- [7] L. Cucu and Y. Sorel. Non-preemptive scheduling algorithms and schedulability conditions for real-time systems with precedence and latency constraints. Research Report RR-5403, INRIA, Rocquencourt, France, 2004.
- [8] J.H.M. Korst, E.H.L. Aarts, and J.K. Lenstra. Scheduling periodic tasks. *INFORMS Journal on Computing* 8, 1996.
- [9] P. Meumeu and Y. Sorel. Non-schedulability conditions for off-line scheduling of real-time systems subject to precedence and strict periodicity constraints. In *Proceedings of 11th IEEE International Conference on Emerging technologies and Factory Automation, ETFA'06*, WIP, Prague, Czech Republic, September 2006.
- [10] P. Meumeu and Y. Sorel. Schedulability analysis with exact number of preemptions and no idle time for real-time systems with precedence and strict periodicity constraints. In *Proceedings of 15th International Conference on Real-Time and Network Systems, RTNS'07*, Nancy, France, March 2007.
- [11] Burns A., Tindell K., and Wellings A. Effective analysis for engineering real-time fixed priority schedulers. *IEEE Trans. Software Eng.*, 1995.
- [12] J. Echagüe, I. Ripoll, and A. Crespo. Hard real-time preemptively scheduling with high context switch cost. In *Proceedings of the 7th Euromicro Workshop on Real-Time Systems*, 1995.
- [13] P. Meumeu and Y. Sorel. Extending rate monotonic analysis with exact cost of preemptions for hard real-time systems. In *Proceedings of 19th Euromicro Conference on Real-Time Systems, ECRTS'07*, Pisa, Italy, July 2007.
- [14] C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in a hard-real-time environment. *Journal of the ACM*, 1973.
- [15] J. Goossens. Scheduling of offset free systems. *The Journal of Real-Time Systems*, 2003.
- [16] J. Goossens and Devilliers R. The non-optimality of the monotonic priority assignments for hard real-time offset free systems. *Real-Time Systems*, 1997.
- [17] M. Grenier, J. Goossens, and N. Navet. Near-optimal fixed priority preemptive scheduling of offset free systems. *14th International Conference on Real-time and Network Systems*, Poitiers, 2006.

## Contents

<b>1</b>	<b>Introduction</b>	<b>3</b>
<b>2</b>	<b>Model</b>	<b>4</b>
<b>3</b>	<b>Definitions</b>	<b>6</b>
<b>4</b>	<b>The proposed approach</b>	<b>10</b>
4.1	Scheduling of two operations . . . . .	10
4.2	Scheduling of $n > 2$ tasks . . . . .	13
4.3	Scheduling algorithm . . . . .	14
<b>5</b>	<b>Complexity of the proposed scheduling algorithm</b>	<b>19</b>
<b>6</b>	<b>Conclusion and future work</b>	<b>19</b>



---

Centre de recherche INRIA Paris – Rocquencourt  
Domaine de Voluceau - Rocquencourt - BP 105 - 78153 Le Chesnay Cedex (France)

Centre de recherche INRIA Bordeaux – Sud Ouest : Domaine Universitaire - 351, cours de la Libération - 33405 Talence Cedex  
Centre de recherche INRIA Grenoble – Rhône-Alpes : 655, avenue de l'Europe - 38334 Montbonnot Saint-Ismier  
Centre de recherche INRIA Lille – Nord Europe : Parc Scientifique de la Haute Borne - 40, avenue Halley - 59650 Villeneuve d'Ascq  
Centre de recherche INRIA Nancy – Grand Est : LORIA, Technopôle de Nancy-Brabois - Campus scientifique  
615, rue du Jardin Botanique - BP 101 - 54602 Villers-lès-Nancy Cedex  
Centre de recherche INRIA Rennes – Bretagne Atlantique : IRISA, Campus universitaire de Beaulieu - 35042 Rennes Cedex  
Centre de recherche INRIA Saclay – Île-de-France : Parc Orsay Université - ZAC des Vignes : 4, rue Jacques Monod - 91893 Orsay Cedex  
Centre de recherche INRIA Sophia Antipolis – Méditerranée : 2004, route des Lucioles - BP 93 - 06902 Sophia Antipolis Cedex

---

Éditeur  
INRIA - Domaine de Voluceau - Rocquencourt, BP 105 - 78153 Le Chesnay Cedex (France)  
<http://www.inria.fr>  
ISSN 0249-6399